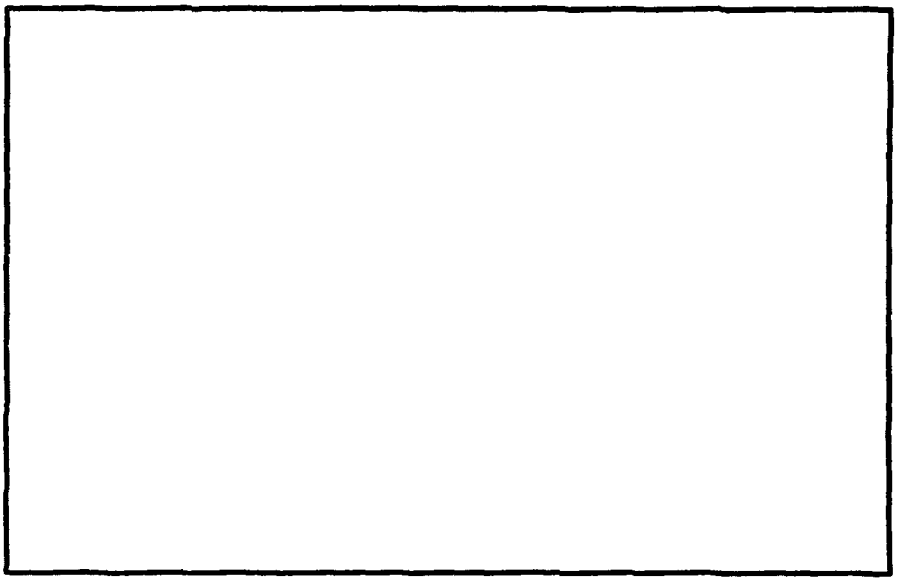
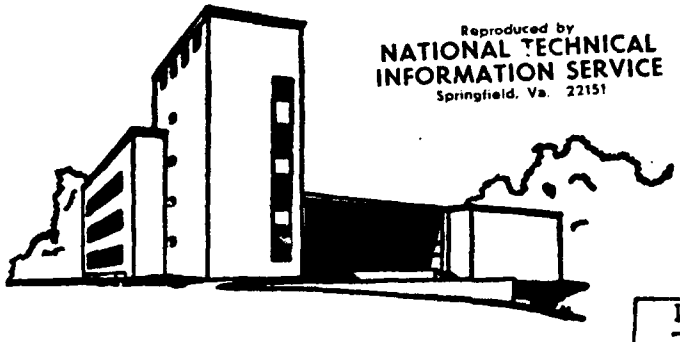


Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

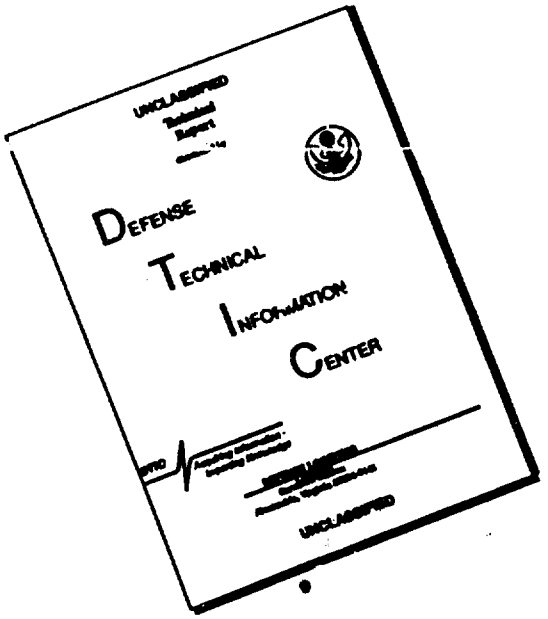
GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



26

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

Graduate School of Industrial Administration Carnegie-Mellon University	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP Not Applicable

3. REPORT TITLE

A LINEAR PROGRAMMING FORMULATION OF A SPECIAL QUADRATIC ASSIGNMENT PROBLEM

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Management Sciences Research Report November 1971

5. AUTHOR(S) (First name, middle initial, last name)

V. J. Bowman
D. A. Pierce
R. Ramsey

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 1971	20	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-67-A-0314-0007 b. PROJECT NO. NR 047-048 c. d.	Management Sciences Research Report No. 277 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) WP - 39-71-2

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale;
its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Logistics and Mathematical Statistics Br. Office of Naval Research Washington, D. C. 20360

13. ABSTRACT

A special quadratic assignment problem is shown to be equivalent to a linear programming problem with n^3 constraints and n^2 variables where n is the number of elements to be assigned. A labeling algorithm similar to that for the linear transportation problem is presented for solving the problem. An example is presented that deals with "triangularizing" input-output matrices.

DD FORM 1 JAN 68 1473 (PAGE 1)

S/N 0101-807-6801

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Quadratic Assignment						
Linear Programming						
Labeling Algorithm						
Permutation Polyhedra						
Matrix Generation						
Dual Simplex Methods						
Input-Output Matrix						

Management Sciences Research Report No. 277

A LINEAR PROGRAMMING FORMULATION OF A SPECIAL QUADRATIC ASSIGNMENT PROBLEM

by

V. J. Bowman*

D. A. Pierce**

F. Ramsey**

November 1971

* Graduate School of Industrial Administration, Carnegie-Mellon University and
Department of Industrial Engineering, University of Pittsburgh

** Department of Statistics
Oregon State University

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract N00014-67-A-0314-0007 NR 047-048 with the U. S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

A LINEAR PROGRAMMING FORMULATION OF A SPECIAL QUADRATIC ASSIGNMENT PROBLEM

Abstract

A special quadratic assignment problem is shown to be equivalent to a linear programming problem with n^3 constraints and n^2 variables where n is the number of elements to be assigned. A labeling algorithm similar to that for the linear transportation problem is presented for solving the problem. An example is presented that deals with "triangularizing" input-output matrices.

Introduction

This paper shows that a special quadratic assignment problem can be reformulated as a linear programming problem.

The general quadratic assignment problem can be written as

$$\begin{aligned} \max \quad & T \cdot Y'AY \\ \text{st} \quad & \sum_i y_{ij} = 1 \\ & \sum_j y_{ij} = 1 \qquad (1) \\ & y_{ij} = 0,1 \end{aligned}$$

where the matrix dot product is defined as $T \cdot C = \sum_{(i,j)} t_{ij} c_{ij}$, i.e., is sum of products of all paired indices, see Lawler [3]. In Section 1, we show that for a special form of T, (1) can be reformulated as a linear programming problem, i.e., all extreme points of a linear set of constraints are solutions to (1) and vice-versa. In Section 2, we develop a labeling algorithm for solving the linear programming formulation. In Section 3 we discuss an application of this formulation to the triangularization of input-output matrices.

Section 1: Linear Programming Equivalent

Consider the quadratic assignment problem (1) where T is an upper triangular matrix of ones, i.e.,

$$T = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 \end{pmatrix}$$

This quadratic assignment problem can be viewed as finding a permutation $P = (r_1, \ldots r_n)$ of the rows and columns of A such that the sum of the above diagonal elements is maximized. That is, if $f_A(P)$ is the value of (1) for permutation P then

$$f_A(P) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{r_i, r_j} \quad .$$

The relationship of the permutation P to the variables Y in (1) is that Y is the permutation matrix of P, that is

$$y_{ij} = \begin{cases} 0 & i \neq r_j \\ 1 & i = r_j \end{cases}$$

The structure of the problem yields the following easily verified properties:

Property 1: For any permutation either a_{ij} or a_{ji}, $i \neq j$ must appear above the diagonal so that either a_{ij} or a_{ji} is in the function $f_A(P)$.

Property 2: If A is a symmetric matrix, then $f_A(P)$=constant for all P.

Property 3: If $A^*=A+B$ then $f_{A^*}(P) = f_A(P) + f_B(P)$ for all P.

We can assume by Properties 2 and 3 that $a_{ij} \geq 0$ for all i and j. Let us consider the graph G = (X,U) where X is a set of n vertices and U is the set of directed arcs $U = \{(i,j) \quad i \neq j\}$. We will associate with each arc (i,j) of G a cost a_{ij}. Let Γ be the set of partial graphs formed with $\frac{n(n-1)}{2}$ arcs of G such that $(i,j) \in G' \in \Gamma \Rightarrow (j,i) \notin G'$. Let $h(G') = \sum_{(i,j) \in G'} a_{ij}$ for $G' \in \Gamma$, $G' = (X',U)$. If we consider the following problem

$$\begin{aligned} \min \quad & h(G') \\ \text{st} \quad & G' \in \Gamma \\ & \text{and } G' \text{ has no circuits,} \end{aligned} \qquad (2)$$

then (1) and (2) are related by the following theorem.

Theorem 1. If z is the optimal solution to (1) and w the optimal solution to (2), then z = w

Proof: In order to prove this theorem we will show that every feasible graph, G', in (2) corresponds to a permutation P' in (1) and that there exists no permutation in (1) for which this correspondence does not hold.

Lemma 1: If $G' \in \Gamma$ then $(i,j) \notin G' \Rightarrow (j,i) \in G'$ for $i \neq j$.

Proof: Consider the sets of arcs $S_{ij} = \{(i,j),(j,i) \mid i > j\}$; There are n(n-1)/2 such sets and by hypothesis only one arc can be drawn from each set in constructing G'. If there exists a set S_{ij} such that $S_{ij} \cap U' = \emptyset$ then $|U'| < n(n-1)/2$ contradicting the face that $G' \in \Gamma$.

Lemma 2: If there exists a circuit in $G' \in \Gamma$ of length $g > 3$ then there exists a circuit of length 3.

Proof: Let $(i_1, i_2, \ldots, i_{q-1}, i_1)$ denote the circuit $G' \in \Gamma$. Now suppose $(i_1, i_{q-2}) \notin G'$ since otherwise we are done. By Lemma 1, $(i_{q-2}, i_1) \in G'$; thus there exists a circuit of length $g-1$. This indirective step can be repeated to prove the lemma.

Let us define the out degree $N(i)$ of a vertex i in G' as $N(i) = \sum_{(i,j) \in G'} 1$

Lemma 3: If $N(i) = N(j)$ $i \neq j$ then G' has a circuit.

Proof: Let $N(i) = N(j_1) = K$ and let $(i,j_1),(i,j_2),\ldots,(i,j_K) \in G'$. Now suppose $(j_1,t) \in G'$ and $t \neq j_i$, $i = 2,\ldots,K$, then $(t,i) \in G'$ by Lemma 1. Therefore $t = j_i$, $i = 2,\ldots,K$, but there are only $K-1$ such arcs and since $N(j_1)=K$ there is one arc (j_1,t) for which $t \neq j_i$, $i = 2,\ldots,K$ and there is a circuit.

Proof of Theorem 1.

Let G' be feasible in (2). Then the nodes of G' can be ordered such that $i < j$ if $N(i) > N(j)$. Let this ordering be $P' = (r_1, r_2, \ldots, r_n)$ i.e., $r_j = k$ if $N(k) = n - j$. Since $0 \leq N(i) \leq n-1, N(i_j) = N(i_{j+1}) + 1$, and thus, P' is a permutation of the n integers $(0,1,\ldots,n-1)$, which is isomorphic to the permutation of the integers $(1,2,\ldots,n)$. Consequently P' is a permutation for (1). Notice that any permutation in (1), introduces the ordering on G' by the out degree values. To show that $z = w$, we note that for the permutation P', $z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{r_i r_j}$ while for the associated graph $G' \in \Gamma$,

$$w = h(G') = \sum_{(i,j) \in G'} a_{ij} = \sum_{i=1}^{n-1} \sum_{j=1}^{n} a_{r_i r_j}.$$

Q.E.D.

Corollary: The optimal solutions to (1) and (2) are related as follows:

$$y_{ij} = \begin{cases} 1 \text{ if } N(i) = n-j \\ 0 \text{ if otherwise} \end{cases}$$

Proof: Obvious from the proof of Theorem 1.

From Lemma 2 we can reformulate (2) as follows:

$$\begin{aligned} \min \quad & h(G') \\ \text{st} \quad & G' \in \Gamma \\ & G' \text{ has no circuits of length 3.} \end{aligned}$$

If we let $x_{ij} = 1$ if $(i,j) \in G'$

$= 0$ otherwise,

then the following equations do not allow circuits of length three:

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \qquad i \neq j \neq k$$

$$x_{ji} + x_{ik} + x_{kj} \leq 2;$$

in addition, Property 1 stipulates $x_{ij} + x_{ji} = 1 \quad i \neq j$. Thus the quadratic assignment problem has been reduced to the following integer linear programming problem

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij}$$

$$\begin{aligned} \text{st} \quad & x_{ij} + x_{jk} + x_{ki} \leq 2 \quad i \neq k \neq j \neq i \qquad (4) \\ & x_{ik} + x_{kj} + x_{ji} \leq 2 \\ & x_{ij} + x_{jk} = 1 \qquad i \neq j \\ & x_{ii} = 0 \text{ for all } i \\ & x_{ij} = 0,1. \end{aligned}$$

We now use the following theorem of Bowman [1] to reduce (4) and correspondingly the quadratic assignment problem (1) to a linear program.

Theorem 2: The extreme points of the polyhedron formed by the constraints

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \qquad i \neq k \neq j \neq i$$

$$x_{ik} + x_{kj} + x_{ji} \leq 2$$

$$x_{ij} + x_{jk} = 1 \qquad i \neq j$$

$$x_{ij} \geq 0$$

$$x_{ii} = 0 \text{ for all } i$$

are all integer.

The result of this theorem is that we can drop the integer constraints to (4) which reduces the problem to a linear programming problem. However, there are $n(n-1)$ variables and $n(n-1)/2 + n(n-1)(n-2)/3$ constraints which causes problems with solving by conventional L.P. methods as n grows large. In the next section, we develop a labeling technique to solve (4).

Section 2: Algorithm

In this section we describe a labelling type algorithm for solving the linear programming problem (4). This algorithm corresponds to a dual simplex routine, but the amount of computer storage required is smaller than the standard linear programming requirements.

We first note that the constraint $x_{ij} + x_{ji} = 1$ can be replaced by upper bound constraints on one of the variables. Therefore, let us reformulate (4) as

$$\begin{aligned} \max \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (a_{ij} - a_{ji}) x_{ij} \\ & x_{ij} + x_{jk} - x_{ik} \leq 1 \qquad i < j < k \qquad (5) \\ & -x_{ij} - x_{jk} + x_{ik} \leq 0 \qquad i < j < k \\ & 0 \leq x_{ij} \leq 1 \qquad i < j. \end{aligned}$$

We have thus reduced the problem to one involving $\frac{n(n-1)}{2}$ variables, but there are still $n(n-1)(n-2)/3$ constraints excluding the upper bounds. This, of course, implies solving the dual of (5). Let (5) have the matrix representation

$$\begin{aligned} & \max \quad cx \\ & \text{s.t. } Bx \leq e \\ & \text{and } -Bx \leq 0 \\ & \quad 0 \leq x \leq 1 \end{aligned}$$

then the dual to (5) is

$$\begin{aligned} & \min \quad ew + ev \\ & \text{s.t. } B'w - B'u + Iv \geq c \\ & \quad w,u,v, \geq 0 \end{aligned}$$

or

$$\begin{aligned} &\min \quad ew + ev \\ &\text{s.t.} \quad -B'w + B'u - Iv + Is = -c \qquad (6) \\ &\qquad s,w,u,v, \geq 0 \end{aligned}$$

Now since the columns of B' correspond to the constraints of (5) every column can be denoted by a triple (i,j,k) where $i < j < k$. Of course, identifying a column of B' is equivalent to identifying variables in (6) and we therefore, use the following notation:

(i,j,k) denotes variable u_{ijk} $\quad i < j < k$

and (7a)

$\overline{(i,j,k)}$ denotes variable w_{ijk} $\quad i < j < k.$

In a similar manner we let

$(0,i,j)$ denote variable s_{ij} $\quad i < j$

and (7b)

$\overline{(0,i,j)}$ denote variable v_{ij} $\quad i < j.$

Observe that the variable designation also allows one to generate the appropriate column representation in (6). Because of this fact we will develop an algorithm that keeps track of the basis elements of (6), but not their matrix inverse. Then when we decide to enter a new variable, we will find its representation by a labelling algorithm. This is quite similar to the approach used in transportation algorithms.

In order to determine incoming variables (or optimality) we will use the fact that if we know the dual solution associated with a

basis of (6), i.e., values for x_{ij}, then the reduced costs for w and u are the slacks of the respective constraints for $x_{ij} + x_{jk} - x_{jk} \leq 1$ and $-x_{ij} - x_{jk} + x_{ik} \leq 0$ and the reduced costs for v and s are respectively $1-x_{ij}$ and x_{ij}. When these reduced costs are all positive we are optimal, otherwise we introduce the appropriate variable into the basis. In order to understand the following algorithm we again point out the variable designations (7a) and (7b) allow for the generation of the appropriate column of (6) and thus, there is no need to ever have the complete representation of (6).

We denote as follows various storage units used in the algorithm.

X: Upper triangular matrix whose values x_{ij} correspond to the solution of (5) associated with a basis of (6) i.e., they are the reduced costs associated with variable s.

C: Right-hand side values associated with current basis of (6).

B: Upper triangular matrix b_{ij} is the identification of the basic variable associated with row (i,j) of (6).

S: $n(n-1)/2$ lists denoted S_{ij}. List S_{ij} contains the basic variables that have non-zero entries in row (i,j) of (6).

The other elements used are appropriate temporary storage units.

Step : Initialization

$$c_{ij} = |a_{ij} - a_{ji}|,$$

If $(a_{ij} - a_{ji}) \leq 0$, $x_{ij} = 0$, $b_{ij} = (0,i,j)$, $S_{ij} = \{(0,i,j)\}$

Otherwise, $x_{ij} = 1$, $b_{ij} = \overline{(0,i,j)}$, $S_{ij} = \{\overline{(0,i,j)}\}$

Remark: It is easy to confirm that this produces a feasible basic solution to (6) made up of the variables v and s.

Step 1: Optimality Check

If $0 \leq x_{ij} \leq 1$ for all $i < j$ and

$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1$ for all $i < j < k$, then stop, this solution is optimal.

Remark: Let $v = \{(r,s,t)$ or $(\overline{r,s,t})\}$ be the entering variable corresponding to violation of one of the above restrictions. Note $r = 0$ is permissable. Go to Step 2.

Remark: When $r = 0$, addition on index 0 implies no operation and is used only for notational convenience.

Step 2:

2A: Let $z_{ij} = |S_{ij}|$ (the number of elements in list S_{ij}).

Let $z_{rs} = z_{rs} + 1$, $z_{rt} = z_{rt} + 1$, $z_{st} = z_{st} + 1$.

2b: Reduction Step

2B1: $D = \emptyset$ (D is a list)

2B2: If $z_{ij} = 1$ for any z_{ij}, let $P = S_{ij} - S_{ij} \cap D =$

$= \{(a,b,c)$ or $(\overline{a,b,c})\}$ and go to Step 2B3.

Otherwise go to Step 3.

2B3: $D = D \cup P$ and $z_{ab} = z_{ab} - 1$, $z_{ac} = z_{ac} - 1$ and $z_{bc} = z_{bc} - 1$.

Go to 2B2.

Remark: Step 2B eliminates basic variables that cannot be used to represent (r,s,t) or $(\overline{r,s,t})$ because of having only one non-zero element in a row of the basis matrix that has a zero in the entering column.

Step 3:

3A: Let $E_{ij} = 0$, $i < j$ and $z_{rs} = z_{rs} - 1$, $z_{rt} = z_{rt} - 1$, $z_{st} = z_{st} - 1$.

3B: If $v = (r,s,t)$, $E_{rs} = -1$, $E_{rt} = 1$, $E_{st} = -1$. Go to Step 3C.

Otherwise, if $v = \overline{(r,s,t)}$, $E_{rs} = 1$, $E_{rt} = -1$, $E_{st} = 1$.

Go to 3C.

3C: Let $Q = \{\emptyset\}$ (an ordered list).

3D:

3D1: If $z_{ij} = 1$ and $E_{ij} \neq 0$ for some $i < j$, let $P = S_{ij} - S_{ij} \cap D$.

Let $b_{pq} = P$ and $L_1(b_{pq}) = +1$.

Go to 3E.

3D2: If $E_{ij} = 0$ for all i,j; go to Step 4.

3D3: If $z_{ij} = 0$ for some $E_{ij} \neq 0$. Go to Step 5.

If $z_{ij} \geq 2$ for all $E_{ij} \neq 0$, let P = some element of $S_{ij} - S_{ij} \cap D$ for $E_{ij} \neq 0$. Let $b_{pq} = P$. $L_1(b_{pq}) = 0$.

Go to 3E.

3E:

3E1: If $b_{pq} = (a,b,c)$ then either $(a,b) = (i,j)$ and set

$L_2(b_{pq}) = \text{sign } (-E_{ij})$ or $(a,c) = (i,j)$ and set

$L_2(b_{pq}) = \text{sign } (E_{ij})$ or $(b,c) = (i,j)$ and set

$L_2(b_{pq}) = \text{sign } (-E_{ij})$;

$E_{ab} = E_{ab} + L_2(b_{pq}) \cdot 1$

$E_{ac} = E_{ac} - L_2(b_{pq}) \cdot 1$

$E_{bc} = E_{bc} + L_2(b_{pq}) \cdot 1$ Go to 3F.

3E2: Otherwise if $b_{pq} = \overline{(a,b,c)}$ then either $(a,b) = (i,j)$ and set $L_2(b_{pq}) = \text{sign } (E_{ij})$ or $(a,c) = (i,j)$ and set

$L_2(b_{pq}) = \text{sign } (-E_{ij})$ or $(b,c) = (i,j)$ and set

$L_2(b_{pq}) = \text{sign } (E_{ij})$;

$E_{ab} = E_{ab} - L_2(b_{pq}) \cdot 1$

$E_{ac} = E_{ac} + L_2(b_{pq}) \cdot 1$

$E_{bc} = E_{bc} - L_2(b_{pq}) \cdot 1$ Go to 3F.

3F: $z_{ab} = z_{ab} - 1$, $z_{ac} = z_{ac} - 1$, $z_{bc} = z_{bc} - 1$, $D = D \cup P$
$Q = Q \cup (b_{pq}, L_1(b_{pq}), L_2(b_{pq}))$ Go to 3D.

Remark: Step 3 finds the representation of the incoming column v in terms of the present basis. L_1 is a label that is one if there is only one non-zero entry in a row which has a non-zero value in v or its subsequent representation. Step 3D1 always tries to force a permanent labelling since this is a forced relationship.

L_2 is a label that gives the sign of the basis element b_{pq} in its representation of v. Since the set is unimodular, we know the coefficient of a basis vector is 0 or ± 1.

Finally, we note that 3D2 stops our labelling procedure when we have found a linear combination of the basis vectors that yields v.

Step 4: Change of Basis.

4A: Let Q^+ be the set of Q such that $L_2(b_{ij}) = +$ for $b_{ij} \in Q$ and Q^- be the set of Q such that $L_2(b_{ij}) = -$ for $b_{ij} \in Q$. Let $\min_{b_{ij} \in Q^+} c_{ij} = c_{pq}$ and $W = b_{pq}$.

$c_{ij} = c_{ij} + c_{pq}$ for all $b_{ij} \in Q^-$ and
$c_{ij} = c_{ij} - c_{pq}$ for all $b_{ij} \in Q^+$ $(i,j) \neq (p,q)$.

$b_{pq} = v$ and $S_{rs} = S_{rs} \cup v$, $S_{rt} = S_{rt} \cup v$, $S_{st} = S_{st} \cup v$
and if $W = (a,b,c)$ or $\overline{(a,b,c)}$ we have $S_{ab} = S_{ab} - W$, $S_{ac} = S_{ac} - W$ and $S_{bc} = S_{bc} - W$.

4B: For $(0,i,j) \in B$, set $x_{ij} = 0$, for $\overline{(0,i,j)} \in B$ set $x_{ij} = 1$.
Go to 4C.

4C: For $(i,j,k) \in B$, $i \neq 0$ set, $x_{ij} + x_{jk} - x_{ik} = 0$ and for $\overline{(i,j,k)} \in B$ set $x_{ij} + x_{jk} - x_{ik} = 1$. Solve these constraints with the conditions of 4B to obtain new X matrix. Go to Step 1.

Step 5: Back-tracking.

5A: Back-track in list Q to the last element labelled 0 (temporary) say b_{mn}. All elements encountered in the back-tracking prior to b_{mn} say b_{pq} must have $L_1(b_{pq}) = +1$ and are freed i.e., for all p,q. Perform 5A1, 5A2, and 5A3.

5A1: If $b_{pq} = (a,b,c)$, then

$E_{ab} = E_{ab} - L_2(b_{pq}) \cdot 1$

$E_{ac} = E_{ac} + L_2(b_{pq}) \cdot 1$

$E_{bc} = E_{bc} - L_2(b_{pq}) \cdot 1$. Go to Step 5A3.

5A2: Otherwise, if $b_{pq} = \overline{(a,b,c)}$, then

$E_{ab} = E_{ab} + L_2(b_{pq}) \cdot 1$

$E_{ac} = E_{ac} - L_2(b_{pq}) \cdot 1$

$E_{bc} = E_{bc} + L_2(b_{pq}) \cdot 1$. Go to Step 5A3.

5A3: $D = D - b_{pq}$, $Q = Q - (b_{pq}, L_1(b_{pq}), L_2(b_{pq}))$

$z_{ab} = z_{ab} + 1$, $z_{ac} = z_{ac} + 1$, $z_{bc} = z_{bc} + 1$.

5B: $L_1(b_{mn}) = +1$. Go to 3D.

Remark: Note that the only change for b_{mn} is that it is permanently labelled rather then temporary. This makes sure that it will be freed when we back-track to some point prior to its assignment.

While this algorithm may seem quite involved compared to simple simplex operations, one should note that its advantage comes in its small core requirement compared to a linear programming approach. If we assume that the original matrix is generated rather than stored,

our core requirement for simplex solution of (6) is essentially that of the basis inverse. This is $(\frac{n(n-1)}{2})^2$ or on the order of n^4. For the above algorithm, one needs to store $\frac{n(n-1)}{2}$ elements for X,C,B,Z,E, and 2 labels and a maximum of $\frac{3n(n-1)}{2}$ for S, that is approximately $5n(n-1)$. For large n this savings is considerable. For n = 20 the L.P. may require over 36,000 storage spaces while the proposed algorithm requires roughly 1900.

In the next section, we discuss the application of this algorithm to a quadratic assignment problem involving the triangularization of Input-Output matrices.

Section 3: Example - Triangularization Input-Output Matrices

The permutation of rows and columns of an input-output matrix, denoted A, that maximize the sum of the above diagonal elements provides a hierarchical listing of the sectors of the economy as "users" and "suppliers".

For the economic interpretations and a heuristic method for triangularization, see Simpson and Tsukui [6]. For an implicit enumeration algorithm, see [5].

It is easily verified that the problem of permuting the rows and columns of A makes it a quadratic programming problem and since we are concerned with the sum of the above diagonal elements we have the T matrix as in Section 1.

The following example is a 5x5 subset of an original 37x37 input-output matrix presented by Leontief [4].

Agric. and Fish	2453	3896	2195	15	317
Food and Kindred Prod.	538	1427	61	8	0
Textile Mill	14	0	1321	2913	0
Apparel	9	50	0	1471	0
Lumber and Food	34	25	20	0	1817

Step 0:

$$c = \begin{pmatrix} 3358 & 2181 & 6 & 283 \\ & 61 & 42 & 25 \\ & & 2913 & 20 \\ & & & 0 \end{pmatrix} \quad x = \begin{pmatrix} 1 & 1 & 1 & 1 \\ & 1 & 0 & 0 \\ & & 1 & 0 \\ & & & 0 \end{pmatrix}$$

$$B = \begin{pmatrix} \overline{(0,1,2)} & \overline{(0,1,3)} & \overline{(0,1,4)} & \overline{(0,1,5)} \\ & \overline{(0,2,3)} & (0,2,4) & (0,2,5) \\ & & \overline{(0,3,4)} & (0,3,5) \\ & & & (0,4,5) \end{pmatrix}$$

$S = B$.

Step 1: $x_{23} + x_{34} - x_{24} = 2$ and $v = \overline{(2,3,4)}$

Step 2:

2A:
$$z = \begin{pmatrix} 1 & 1 & 1 & 1 \\ & 2 & 2 & 1 \\ & & 2 & 1 \\ & & & 1 \end{pmatrix}$$

2B1: $D = \emptyset$

2B2: $z_{1,2} = 1$, $P = \overline{(0,1,2)}$

2B3: $D = \{\overline{(0,1,2)}\}$

$$z = \begin{pmatrix} 0 & 1 & 1 & 1 \\ & 2 & 2 & 1 \\ & & 2 & 1 \\ & & & 1 \end{pmatrix}$$

2B2: $z_{1,3} = 1$, $P = \overline{(0,1,3)}$

2B3: $D = \{\overline{(0,1,2)}, \overline{(0,1,3)}\}$

$$z = \begin{pmatrix} 0 & 0 & 1 & 1 \\ & 2 & 2 & 1 \\ & & 2 & 1 \\ & & & 1 \end{pmatrix}$$

2B2: $z_{1,4} = 1$, $P = \overline{(0,1,4)}$

2B3: $D = \{\overline{(0,1,2)}, \overline{(0,1,3)}, \overline{(0,1,4)}$

$$z = \begin{pmatrix} 0 & 0 & 0 & 1 \\ & 2 & 2 & 1 \\ & & 2 & 1 \\ & & & 1 \end{pmatrix}$$

2B2 and 2B3 repeat for $z_{1,5}, z_{2,5}, z_{3,5}$, and $z_{4,5}$ ending with

2B3: $D = \{\overline{(0,1,2)}, \overline{(0,1,3)}, \overline{(0,1,4)}, \overline{(0,1,5)}, (0,2,5), (0,3,5), (0,4,5)\}$

$$z = \begin{matrix} 0 & 0 & 0 & 0 \\ & 2 & 2 & 0 \\ & & 2 & 0 \\ & & & 0 \end{matrix}$$

3A: $$E = \begin{matrix} 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{matrix} \qquad z = \begin{matrix} 0 & 0 & 0 & 0 \\ & 1 & 1 & 0 \\ & & 1 & 0 \\ & & & 0 \end{matrix}$$

3B: $$E = \begin{matrix} 0 & 0 & 0 & 0 \\ & 1 & -1 & 0 \\ & & 1 & 0 \\ & & & 0 \end{matrix}$$

3C: $Q = \emptyset$

3D1: $z_{2,3} = 1$ and $E_{2,3} \neq 0$ $\quad P = \overline{(0,2,3)}$.

$L_1(b_{2,3}) = +1$

3E: $L_2(b_{2,3}) = +$ $\quad Q = \{(b_{2,3}, 1, +)\}$

$$E = \begin{matrix} 0 & 0 & 0 & 0 \\ & 0 & -1 & 0 \\ & & 1 & 0 \\ & & & 0 \end{matrix} \qquad z = \begin{matrix} 0 & 0 & 0 & 0 \\ & 0 & 1 & 0 \\ & & 1 & 0 \\ & & & 0 \end{matrix}$$

3D1: $z_{2,4} = 1$ and $E_{2,4} \neq 0$ $\quad p = (0,2,4)$

3E: $L_2(b_{2,4}) = +$, $Q = \{(b_{2,3}, 1, +), (b_{2,4}, 1, +)\}$

$$E = \begin{matrix} 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & & 1 & 0 \\ & & & 0 \end{matrix} \qquad z = \begin{matrix} 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & & 1 & 0 \\ & & & 0 \end{matrix}$$

3D1: $z_{3,4} = 1 \quad E_{3,4} \neq 0 \quad P = \overline{(0,3,4)}$

3E: $L_2(b_{3,4}) = + \quad Q = \{(b_{2,3},1,+),(b_{2,4},1,+),(b_{3,4},1,+)\}$

$$E = \begin{pmatrix} 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{pmatrix} \qquad z = \begin{pmatrix} 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{pmatrix}$$

3D2: $E = 0$

4A: $Q^+ = Q, \; Q^- = \emptyset$

$c_{pq} = \min(61,42,2913) = c_{2,4} \qquad W = (0,2,4)$

$$c = \begin{pmatrix} 3358 & 2181 & 6 & 283 \\ & 19 & 42 & 25 \\ & & 2871 & 20 \\ & & & 0 \end{pmatrix}$$

$v_{2,4} = \overline{(2,3,4)}$

$s_{2,3} = \{\overline{(0,2,3)},\overline{(2,3,4)}\}, \; s_{2,4} = \{\overline{(2,3,4)}\}, \; s_{3,4} = \{\overline{(0,3,4)},\overline{(2,3,4)}\}$

4B: $$x = \begin{pmatrix} 1 & 1 & 1 & 1 \\ & 1 & x & 0 \\ & & 1 & 0 \\ & & & 0 \end{pmatrix}$$

4C: $x_{2,3} + x_{3,4} - x_{2,4} = 1 \qquad x_{2,3} = 1, \; x_{3,4} = 1 \Rightarrow x_{2,4} = 1$

$$x = \begin{pmatrix} 1 & 1 & 1 & 1 \\ & 1 & 1 & 0 \\ & & 1 & 0 \\ & & & 0 \end{pmatrix}$$

1: All conditions satisfied. Solucion optimal.

The optimal solution has

$$x^* = \begin{pmatrix} x & 1 & 1 & 1 & 1 \\ 0 & x & 1 & 1 & 0 \\ 0 & 0 & x & 1 & 0 \\ 0 & 0 & 0 & x & 0 \\ 0 & 1 & 1 & 1 & x \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 1 \\ 0 \\ 3 \end{pmatrix}$$

so that the optimal arrangement is (1,5,2,3,4) i.e.,

Agric. & Fish	2453	317	3896	2195	15
Lumber & Food	34	1817	25	20	0
Food & Kindred Prod.	538	0	1427	61	8
Textile Mills	14	0	0	1321	2913
Apparel	9	0	50	0	1471

or in reduced form

x	283	3358	2181	6
0	x	25	20	0
0	0	x	61	0
0	0	0	x	2913
0	0	42	0	x

While the algorithm may seem quite cumbersome for such a small problem, efficient use of bit processing routines and proper programming techniques greatly simplify the updating procedures.

Conclusion

We have shown that a special form of the quadratic assignment problem can be reformulated as a linear programming problem, and that because of the structure of the linear program an algorithm can be developed that parallels the simplex method but at a greatly reduced storage requirement. This might indicate that other quadratic assignment problems may have equivalent linear programming formulations for specific T matrices in the formulation (1). These of course, will not necessarily have to correspond to integer solutions as in the case of the x_{ij} in the above problem, but rather, may fall in the category of searching for appropriate permutation polyhedra which are discussed by Bowman [1]. One should further note that the algorithm of Section 2 is really dependent on the constraint set of (5) and not its objective function and thus, is applicable to any problems where this constraint set arises whether or not it comes from the Quadratic Assignment problem (1). For a discussion of this constraint set in the context of transitivity, see Bowman and Colantoni [2].

References

1. Bowman, V.J., "Permutation Polyhedra" forthcoming in SIAM Journal on Applied Mathematics.

2. Bowman, V.J. and C.S. Colantoni, "Majority Rule Under Transitivity Constraints", Working Paper #89-70-1, GSIA, Carnegie-Mellon University.

3. Lawler, E., "The Quadratic Assignment Problem, Management Science 9, (1963), pp. 586-599.

4. Leontief, W., "Input-Output Economics", Scientific American 185, (1951), pp. 15-21.

5. Ramsey, F.L., D.A. Pierce and V.J. Bowman, "Triangularization of Input-Output Matrices", Technical Report #16, Department of Statistics, Oregon State University, 1969.

6. Simpson, D. and J. Tsukui, "The Fundamental Structure of Input-Output Tables, An International Comparison", Review of Economics and Statistics 47, (1965), pp. 434-446.